System on a chip

A **system on a chip** (**SoC**; /ˌɛsˌoʊˈsiː/ *es-oh-SEE* or /sɒk/ *sock*[nb 1]) is an integrated circuit (also known as a "chip") that integrates all or most components of a computer or other electronic system. These components almost always include a central processing unit (CPU), memory, input/output ports and secondary storage, often alongside other components such as radio modems and a graphics processing unit (GPU) – all on a single substrate or microchip.[1] It may contain digital, analog, mixed-signal, and often radio frequency signal processing functions (otherwise it is considered only an application processor).

The Raspberry Pi uses a system on a chip as an almost fully contained microcomputer. This SoC does not contain any kind of data storage, which is common for a microprocessor SoC.

Higher-performance SoCs are often paired with dedicated and physically separate memory and secondary storage (almost always LPDDR and eUFS or eMMC, respectively) chips, that may be layered on top of the SoC in what's known as a package on package (PoP) configuration, or be placed close to the SoC. Additionally, SoCs may use separate wireless modems.[2]

SoCs are in contrast to the common traditional motherboard-based PC architecture, which separates components based on function and connects them through a central interfacing circuit board.[nb 2] Whereas a motherboard houses and connects detachable or replaceable components, SoCs integrate all of these components into a single integrated circuit. An SoC will typically integrate a CPU, graphics and memory interfaces,[nb 3] hard-disk and USB connectivity,[nb 4] random-access and read-only memories and secondary storage and/or their controllers on a single circuit die, whereas a motherboard would connect these modules as discrete components or expansion cards.

An SoC integrates a microcontroller, microprocessor or perhaps several processor cores with peripherals like a GPU, Wi-Fi and cellular network radio modems, and/or one or more coprocessors. Similar to how a microcontroller integrates a microprocessor with peripheral circuits and memory, an SoC can be seen as integrating a microcontroller with even more advanced peripherals.

More tightly integrated computer system designs improve performance and reduce power consumption as well as semiconductor die area than multi-chip designs with equivalent functionality. This comes at the cost of reduced replaceability of components. By definition, SoC designs are fully or nearly fully integrated across different component modules. For these reasons, there has been a general trend towards tighter integration of components in the computer hardware industry, in part due to the influence of SoCs and lessons learned from the mobile and embedded computing markets. SoCs can be viewed as part of a larger trend towards embedded computing and hardware acceleration.

SoCs are very common in the mobile computing (such as in smartphones and tablet computers) and edge computing markets.[3][4] They are also commonly used in embedded systems such as WiFi routers and the Internet of Things.

Contents

Types

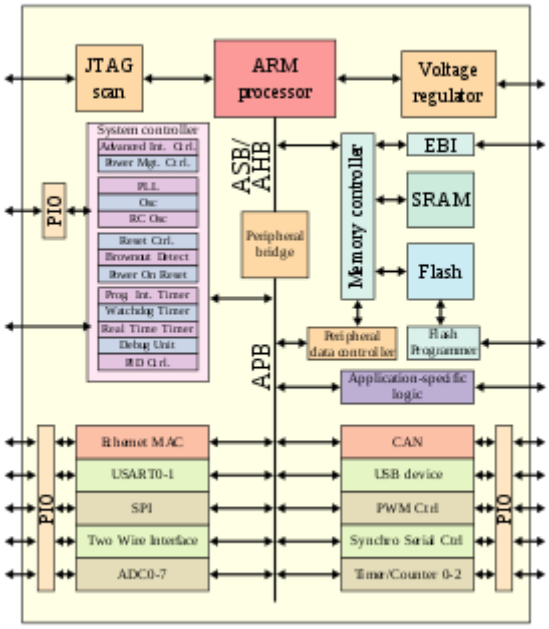
In general, there are four distinguishable types of SoCs:

- SoCs built around a microcontroller,
- SoCs built around a microprocessor, often found in mobile phones;

- Specialized application-specific integrated circuit SoCs designed for specific applications that do not fit into the above two categories, and
- Programmable SoCs (PSoC), where most functionality is fixed but some functionality is reprogrammable in a manner analogous to a field-programmable gate array.

Applications

SoCs can be applied to any computing task. However, they are typically used in mobile computing such as tablets, smartphones, smartwatches and netbooks as well as embedded systems and in applications where previously microcontrollers would be used.



Microcontroller-based system on a chip

Embedded systems

Where previously only microcontrollers could be used, SoCs are rising to prominence in the embedded systems market. Tighter system integration offers better reliability and mean time between failure, and SoCs offer more advanced functionality and computing power than microcontrollers.[5] Applications include AI acceleration, embedded machine vision,[6] data collection, telemetry, vector processing and ambient intelligence. Often embedded SoCs target the internet of things, industrial internet of things and edge computing markets.

AMD Am286ZX/LX, SoC based on Intel 80286

Mobile computing

Mobile computing based SoCs always bundle processors, memories, on-chip caches, wireless networking capabilities and often digital camera hardware and firmware. With increasing memory sizes, high end SoCs will often have no memory and flash storage and instead, the memory and flash memory will be placed right next to, or above (package on package), the SoC.[7] Some examples of mobile computing SoCs include:

- Samsung Electronics: list, typically based on ARM
  - Exynos, used mainly by Samsung's Galaxy series of smartphones
- Qualcomm:
  - Snapdragon (list), used in many LG, Xiaomi, Google Pixel, HTC and Samsung Galaxy smartphones. In 2018, Snapdragon SoCs are being used as the backbone of laptop computers running Windows 10, marketed as "Always Connected PCs".[8][9]

Personal computers

In 1992, Acorn Computers produced the A3010, A3020 and A4000 range of personal computers with the ARM250 SoC. It combined the original Acorn ARM2 processor with a memory controller (MEMC), video controller (VIDC), and I/O controller (IOC). In previous Acorn ARM-powered computers, these were four discrete chips. The ARM7500 chip was their second-generation SoC, based on the ARM700, VIDC20 and IOMD controllers, and was widely licensed in embedded devices such as set-top-boxes, as well as later Acorn personal computers.

SoCs are being applied to mainstream personal computers as of 2018.[8] They are particularly applied to laptops and tablet PCs. Tablet and laptop manufacturers have learned lessons from embedded systems and smartphone markets about reduced power consumption, better performance and reliability from tighter integration of hardware and firmware modules, and LTE and other wireless network communications integrated on chip (integrated network interface controllers).[10]

ARM-based:

- Qualcomm Snapdragon[9]
- ARM250
- ARM7500(FE)
- Apple M1

x86-based:

- Intel Core CULV

Structure

An SoC consists of hardware functional units, including microprocessors that run software code, as well as a communications subsystem to connect, control, direct and interface between these functional modules.

Functional components

Processor cores

An SoC must have at least one processor core, but typically an SoC has more than one core. Processor cores can be a microcontroller, microprocessor (μP),[11] digital signal processor (DSP) or application-specific instruction set processor (ASIP) core.[12] ASIPs have instruction sets that are customized for an application domain and designed to be more efficient than general-purpose instructions for a specific type of workload. Multiprocessor SoCs have more than one processor core by definition.

Whether single-core, multi-core or manycore, SoC processor cores typically use RISC instruction set architectures. RISC architectures are advantageous over CISC processors for SoCs because they require less digital logic, and therefore less power and area on board, and in the embedded and mobile computing markets, area and power are often highly constrained. In particular, SoC processor cores often use the ARM architecture because it is a soft processor specified as an IP core and is more power efficient than x86.[11]

Memory

SoCs must have semiconductor memory blocks to perform their computation, as do microcontrollers and other embedded systems. Depending on the application, SoC memory may form a memory hierarchy and cache hierarchy. In the mobile computing market, this is common, but in many low-power embedded microcontrollers, this is not necessary. Memory technologies for SoCs include read-only memory (ROM), random-access memory (RAM), Electrically Erasable Programmable ROM (EEPROM) and flash memory.[11] As in other computer systems, RAM can be subdivided into relatively faster but more expensive static RAM (SRAM) and the slower but cheaper dynamic RAM (DRAM). When an SoC has a cache hierarchy, SRAM will usually be used to implement processor registers and cores' L1 caches whereas DRAM will be used for lower levels of the cache hierarchy including main memory. "Main memory" may be specific to a single processor (which can be multi-core) when the SoC has multiple processors, in which case it is distributed memory and must be sent via § Intermodule communication on-chip to be accessed by a different processor.[12] For further discussion of multi-processing memory issues, see cache coherence and memory latency.

Interfaces

SoCs include external interfaces, typically for communication protocols. These are often based upon industry standards such as USB, FireWire, Ethernet, USART, SPI, HDMI, I²C, etc. These interfaces will differ according to the intended application. Wireless networking protocols such as Wi-Fi, Bluetooth, 6LoWPAN and near-field communication may also be supported.

When needed, SoCs include analog interfaces including analog-to-digital and digital-to-analog converters, often for signal processing. These may be able to interface with different types of sensors or actuators, including smart transducers. They may interface with application-specific modules or shields.[nb 5] Or they may be internal to the SoC, such as if an analog sensor is built in to the SoC and its readings must be converted to digital signals for mathematical processing.

Digital signal processors

Digital signal processor (DSP) cores are often included on SoCs. They perform signal processing operations in SoCs for sensors, actuators, data collection, data analysis and multimedia processing. DSP cores typically feature very long instruction word (VLIW) and single instruction, multiple data (SIMD) instruction set architectures, and are therefore highly amenable to exploiting instruction-level parallelism through parallel processing and superscalar execution.[12]:4 DSP cores most often feature application-specific instructions, and as such are typically application-specific instruction-set processors (ASIP). Such application-specific instructions correspond to dedicated hardware functional units that compute those instructions.

Typical DSP instructions include multiply-accumulate, Fast Fourier transform, fused multiply-add, and convolutions.

Other

As with other computer systems, SoCs require timing sources to generate clock signals, control execution of SoC functions and provide time context to signal processing applications of the SoC, if needed. Popular time sources are crystal oscillators and phase-locked loops.

SoC peripherals including counter-timers, real-time timers and power-on reset generators. SoCs also include voltage regulators and power management circuits.

Intermodule communication

SoCs comprise many execution units. These units must often send data and instructions back and forth. Because of this, all but the most trivial SoCs require communications subsystems. Originally, as with other microcomputer technologies, data bus architectures were used, but recently designs based on sparse intercommunication networks known as networks-on-chip (NoC) have risen to prominence and are forecast to overtake bus architectures for SoC design in the near future.[13]

Bus-based communication

Historically, a shared global computer bus typically connected the different components, also called "blocks" of the SoC.[13] A very common bus for SoC communications is ARM's royalty-free Advanced Microcontroller Bus Architecture (AMBA) standard.

Direct memory access controllers route data directly between external interfaces and SoC memory, bypassing the CPU or control unit, thereby increasing the data throughput of the SoC. This is similar to some device drivers of peripherals on component-based multi-chip module PC architectures.

Computer buses are limited in scalability, supporting only up to tens of cores (multicore) on a single chip.[13]:xiii Wire delay is not scalable due to continued miniaturization, system performance does not scale with the number of cores attached, the SoC's operating frequency must decrease with each additional core attached for power to be sustainable, and long wires consume large amounts of electrical power. These challenges are prohibitive to supporting manycore systems on chip.[13]:xiii

Network on a chip

In the late 2010s, a trend of SoCs implementing communications subsystems in terms of a network-like topology instead of bus-based protocols has emerged. A trend towards more processor cores on SoCs has caused on-chip communication efficiency to become one of the key factors in determining the overall system performance and cost.[13]:xiii This has led to the emergence of interconnection networks with router-based packet switching known as "networks on chip" (NoCs) to overcome the bottlenecks of bus-based networks.[13]:xiii

Networks-on-chip have advantages including destination- and application-specific routing, greater power efficiency and reduced possibility of bus contention. Network-on-chip architectures take inspiration from communication protocols like TCP and the Internet protocol suite for on-chip communication,[13] although they typically have fewer network layers. Optimal network-on-chip network architectures are an ongoing area of much research interest. NoC architectures range from traditional distributed computing network topologies such as torus, hypercube, meshes and tree networks to genetic algorithm scheduling to randomized algorithms such as random walks with branching and randomized time to live (TTL).

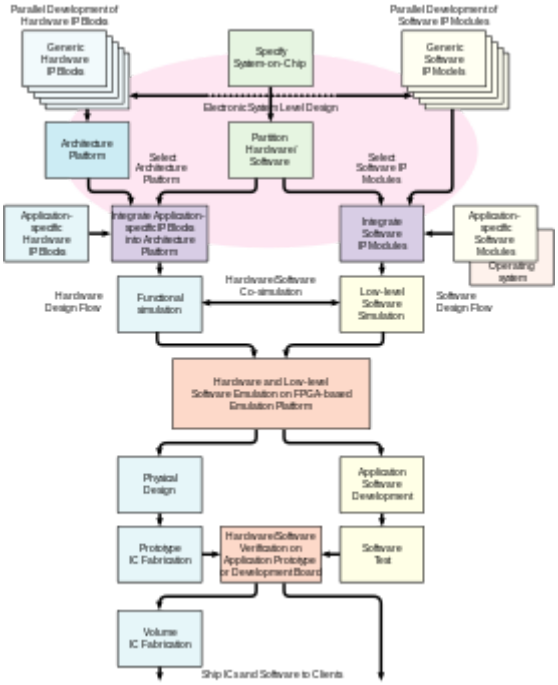
Many SoC researchers consider NoC architectures to be the future of SoC design because they have been shown to efficiently meet power and throughput needs of SoC designs. Current NoC architectures are two-dimensional. 2D IC design has limited floorplanning choices as the number of cores in SoCs increase, so as three-dimensional integrated circuits (3DICs) emerge, SoC designers are looking towards building three-dimensional on-chip networks known as 3DNoCs.[13]

Design flow

A system on a chip consists of both the hardware, described in § Structure, and the software controlling the microcontroller, microprocessor or digital signal processor cores, peripherals and interfaces. The design flow for an SoC aims to develop this hardware and software at the same time, also known as architectural co-design. The design flow must also take into account optimizations (§ Optimization goals) and constraints.

Most SoCs are developed from pre-qualified hardware component IP core specifications for the hardware elements and execution units, collectively "blocks", described above, together with software device drivers that may control their operation. Of particular importance are the protocol stacks that drive industry-standard interfaces like USB. The hardware blocks are put together using computer-aided design tools, specifically electronic design automation tools; the software modules are integrated using a software integrated development environment.

SoCs components are also often designed in high-level programming languages such as C++, MATLAB or SystemC and converted to RTL designs through high-level synthesis (HLS) tools such as C to HDL or flow to HDL.[14] HLS products called "algorithmic synthesis" allow designers to use C++ to model and synthesize system, circuit, software and verification levels all in one high level language commonly known to computer engineers in a manner independent of time scales, which are typically specified in HDL.[15] Other components can remain software and be compiled and embedded onto soft-core processors included in the SoC as modules in HDL as IP cores.



SoC design flow

Once the architecture of the SoC has been defined, any new hardware elements are written in an abstract hardware description language termed register transfer level (RTL) which defines the circuit behavior, or synthesized into RTL from a high level language through high-level synthesis. These elements are connected together in a hardware description language to create the full SoC design. The logic specified to connect these components and convert between possibly different interfaces provided by different vendors is called glue logic.

Design verification

Chips are verified for validation correctness before being sent to a semiconductor foundry. This process is called functional verification and it accounts for a significant portion of the time and energy expended in the chip design life cycle, often quoted as 70%.[16][17] With the growing complexity of chips, hardware verification languages like SystemVerilog, SystemC, e, and OpenVera are being used. Bugs found in the verification stage are reported to the designer.

Traditionally, engineers have employed simulation acceleration, emulation or prototyping on reprogrammable hardware to verify and debug hardware and software for SoC designs prior to the finalization of the design, known as tape-out. Field-programmable gate arrays (FPGAs) are favored for prototyping SoCs because FPGA prototypes are reprogrammable, allow debugging and are more flexible than application-specific integrated circuits (ASICs).[18][19]

With high capacity and fast compilation time, simulation acceleration and emulation are powerful technologies that provide wide visibility into systems. Both technologies, however, operate slowly, on the order of MHz, which may be significantly slower – up to 100 times slower – than the SoC's operating frequency. Acceleration and emulation boxes are also very large and expensive at over US$1 million.

FPGA prototypes, in contrast, use FPGAs directly to enable engineers to validate and test at, or close to, a system's full operating frequency with real-world stimuli. Tools such as Certus[20] are used to insert probes in the FPGA RTL that make signals available for observation. This is used to debug hardware, firmware and software interactions across multiple FPGAs with capabilities similar to a logic analyzer.

In parallel, the hardware elements are grouped and passed through a process of logic synthesis, during which performance constraints, such as operational frequency and expected signal delays, are applied. This generates an output known as a netlist describing the design as a physical circuit and its interconnections. These netlists are combined with the glue logic connecting the components to produce the schematic description of the SoC as a circuit which can be printed onto a chip. This process is known as place and route and precedes tape-out in the event that the SoCs are produced as application-specific integrated circuits (ASIC).

Optimization goals

SoCs must optimize power use, area on die, communication, positioning for locality between modular units and other factors. Optimization is necessarily a design goal of SoCs. If optimization was not necessary, the engineers would use a multi-chip module architecture without accounting for the area utilization, power consumption or performance of the system to the same extent.

Common optimization targets for SoC designs follow, with explanations of each. In general, optimizing any of these quantities may be a hard combinatorial optimization problem, and can indeed be NP-hard fairly easily. Therefore, sophisticated optimization algorithms are often required and it may be practical to use approximation algorithms or heuristics in some cases. Additionally, most SoC designs contain multiple variables to optimize simultaneously, so Pareto efficient solutions are sought after in SoC design. Oftentimes the goals of optimizing some of these quantities are directly at odds, further adding complexity to design optimization of SoCs and introducing trade-offs in system design.

For broader coverage of trade-offs and requirements analysis, see requirements engineering.

Targets

Power consumption

SoCs are optimized to minimize the electrical power used to perform the SoC's functions. Most SoCs must use low power. SoC systems often require long battery life (such as smartphones), can potentially spending months or years without a power source needing to maintain autonomous function, and often are limited in power use by a high number of embedded SoCs being networked together in an area. Additionally, energy costs can be high and conserving energy will reduce the total cost of ownership of the SoC. Finally, waste heat from high energy consumption can damage other circuit components if too much heat is dissipated, giving another pragmatic reason to conserve energy. The amount of energy used in a circuit is the integral of power consumed with respect to time, and the average rate of power consumption is the product of current by voltage. Equivalently, by Ohm's law, power is current squared times resistance or voltage squared divided by resistance:

$$P = IV = \frac{V^2}{R} = I^2R$$

SoCs are frequently embedded in portable devices such as smartphones, GPS navigation devices, digital watches (including smartwatches) and netbooks. Customers want long battery lives for mobile computing devices, another reason that power consumption must be minimized in SoCs. Multimedia applications are

often executed on these devices, including video games, video streaming, image processing; all of which have grown in computational complexity in recent years with user demands and expectations for higher-quality multimedia. Computation is more demanding as expectations move towards 3D video at high resolution with multiple standards, so SoCs performing multimedia tasks must be computationally capable platform while being low power to run off a standard mobile battery.[12]:3

Performance per watt

SoCs are optimized to maximize power efficiency in performance per watt: maximize the performance of the SoC given a budget of power usage. Many applications such as edge computing, distributed processing and ambient intelligence require a certain level of computational performance, but power is limited in most SoC environments. The ARM architecture has greater performance per watt than x86 in embedded systems, so it is preferred over x86 for most SoC applications requiring an embedded processor.

Waste heat

SoC designs are optimized to minimize waste heat output on the chip. As with other integrated circuits, heat generated due to high power density are the bottleneck to further miniaturization of components.[21]:1 The power densities of high speed integrated circuits, particularly microprocessors and including SoCs, have become highly uneven. Too much waste heat can damage circuits and erode reliability of the circuit over time. High temperatures and thermal stress negatively impact reliability, stress migration, decreased mean time between failures, electromigration, wire bonding, metastability and other performance degradation of the SoC over time.[21]:2–9

In particular, most SoCs are in a small physical area or volume and therefore the effects of waste heat are compounded because there is little room for it to diffuse out of the system. Because of high transistor counts on modern devices due to Moore's law, oftentimes a layout of sufficient throughput and high transistor density is physically realizable from fabrication processes but would result in unacceptably high amounts of heat in the circuit's volume.[21]:1

These thermal effects force SoC and other chip designers to apply conservative design margins, creating less performant devices to mitigate the risk of catastrophic failure. Due to increased transistor densities as length scales get smaller, each process generation produces more heat output than the last. Compounding this problem, SoC architectures are usually heterogeneous, creating spatially inhomogeneous heat fluxes, which cannot be effectively mitigated by uniform passive cooling.[21]:1

Throughput

SoCs are optimized to maximize computational and communications throughput.

Latency

SoCs are optimized to minimize latency for some or all of their functions. This can be accomplished by laying out elements with proper proximity and locality to each-other to minimize the interconnection delays and maximize the speed at which data is communicated between modules, functional units and memories. In general, optimizing to minimize latency is an NP-complete problem equivalent to the boolean satisfiability problem.

For tasks running on processor cores, latency and throughput can be improved with task scheduling. Some tasks run in application-specific hardware units, however, and even task scheduling may not be sufficient to optimize all software-based tasks to meet timing and throughput constraints.

Methodologies

Systems on chip are modeled with standard hardware verification and validation techniques, but additional techniques are used to model and optimize SoC design alternatives to make the system optimal with respect to multiple-criteria decision analysis on the above optimization targets.

Task scheduling

Task scheduling is an important activity in any computer system with multiple processes or threads sharing a single processor core. It is important to reduce § Latency and increase § Throughput for embedded software running on an SoC's § Processor cores. Not every important computing activity in a SoC is performed in software running on on-chip processors, but scheduling can drastically improve performance of software-based tasks and other tasks involving shared resources.

SoCs often schedule tasks according to network scheduling and randomized scheduling algorithms.

Pipelining

Hardware and software tasks are often pipelined in processor design. Pipelining is an important principle for speedup in computer architecture. They are frequently used in GPUs (graphics pipeline) and RISC processors (evolutions of the classic RISC pipeline), but are also applied to application-specific tasks such as digital signal processing and multimedia manipulations in the context of SoCs.[12]

Probabilistic modeling

SoCs are often analyzed though probabilistic models, Queueing theory § Queueing networks and Markov chains. For instance, Little's law allows SoC states and NoC buffers to be modeled as arrival processes and analyzed through Poisson random variables and Poisson processes.

Markov chains

SoCs are often modeled with Markov chains, both discrete time and continuous time variants. Markov chain modeling allows asymptotic analysis of the SoC's steady state distribution of power, heat, latency and other factors to allow design decisions to be optimized for the common case.

Fabrication

SoC chips are typically fabricated using metal–oxide–semiconductor (MOS) technology.[22] The netlists described above are used as the basis for the physical design (place and route) flow to convert the designers' intent into the design of the SoC. Throughout this conversion process, the design is analyzed with static timing modeling, simulation and other tools to ensure that it meets the specified operational parameters such as frequency, power consumption and dissipation, functional integrity (as described in the register transfer level code) and electrical integrity.

When all known bugs have been rectified and these have been re-verified and all physical design checks are done, the physical design files describing each layer of the chip are sent to the foundry's mask shop where a full set of glass lithographic masks will be etched. These are sent to a wafer fabrication plant to create the SoC dice before packaging and testing.

SoCs can be fabricated by several technologies, including:

- Full custom ASIC
- Standard cell ASIC
- Field-programmable gate array (FPGA)

ASICs consume less power and are faster than FPGAs but cannot be reprogrammed and are expensive to manufacture. FPGA designs are more suitable for lower volume designs, but after enough units of production ASICs reduce the total cost of ownership.[23]

SoC designs consume less power and have a lower cost and higher reliability than the multi-chip systems that they replace. With fewer packages in the system, assembly costs are reduced as well.

However, like most very-large-scale integration (VLSI) designs, the total cost is higher for one large chip than for the same functionality distributed over several smaller chips, because of lower yields and higher non-recurring engineering costs.

When it is not feasible to construct an SoC for a particular application, an alternative is a system in package (SiP) comprising a number of chips in a single package. When produced in large volumes, SoC is more cost-effective than SiP because its packaging is simpler.[24] Another reason SiP may be preferred is waste heat may be too high in a SoC for a given purpose because functional components are too close together, and in an SiP heat will dissipate better from different functional modules since they are physically further apart.

Benchmarks

SoC research and development often compares many options. Benchmarks, such as COSMIC,[25] are developed to help such evaluations.

See also

- List of system-on-a-chip suppliers
- Post-silicon validation
- ARM architecture
- Single-board computer
- System in package
- Network on a chip
- Programmable SoC
- Application-specific instruction set processor (ASIP)
- Platform-based design
- Lab on a chip
- Organ on a chip in biomedical technology
- Multi-chip module
- List of Qualcomm Snapdragon processors - Qualcomm
- Exynos - Samsung

Notes

1. This article uses the convention that SoC is pronounced /ˌɛsˌoʊˈsiː/ *es-oh-SEE*. Therefore, it uses the convention "an" for the indefinite article corresponding to SoC ("**an** SoC"). Other sources may pronounce it as /sɒk/ *sock* and therefore use "**a** SoC".
2. This central board is called the "mother board" for hosting the "child" component cards.
3. The graphics connections (PCI Express) and RAM historically constituted the northbridge of motherboard-backed discrete architectures.
4. The hard disk and USB connectivity historically comprised part of the southbridge of motherboard-backed discrete modular architectures.
5. In embedded systems, "shields" are analogous to expansion cards for PCs. They often fit over a microcontroller such as an Arduino or single-board computer such as the Raspberry Pi and function as peripherals for the device.

References

1. Shah, Agam (January 3, 2017). "7 dazzling smartphone improvements with Qualcomm's Snapdragon 835 chip" (https://www.networkworld.com/article/3154386/7-dazzling-smartphone-improvements-with-qualcomms-snapdragon-835-chip.html). *Network World*.
2. https://arstechnica.com/gadgets/2020/02/qualcomms-snapdragon-x60-promises-smaller-5g-modems-in-2021/?amp=1
3. Pete Bennett, EE Times. "The why, where and what of low-power SoC design (http://www.eetimes.com/document.asp?doc_id=1276973)." December 2, 2004. Retrieved July 28, 2015.
4. Nolan, Stephen M. "Power Management for Internet of Things (IoT) System on a Chip (SoC) Development" (https://www.design-reuse.com/articles/42705/power-management-for-iot-soc-development.html). *Design And Reuse*. Retrieved 2018-09-25.
5. "Is a single-chip SOC processor right for your embedded project?" (https://www.embedded.com/design/mcus-processors-and-socs/4419584/Is-a-single-chip-SOC-processor-right-for-your-embedded-project-). *Embedded*. Retrieved 2018-10-13.
6. "Qualcomm launches SoCs for embedded vision | Imaging and Machine Vision Europe" (https://www.imveurope.com/news/qualcomm-launches-socs-embedded-vision). *www.imveurope.com*. Retrieved 2018-10-13.
7. "Samsung Galaxy S10 and S10e Teardown" (https://www.ifixit.com/Teardown/Samsung+Galaxy+S10+and+S10e+Teardown/120331). *iFixit*. March 6, 2019.
8. "ARM is going after Intel with new chip roadmap through 2020" (https://www.windowscentral.com/arm-going-after-intel-new-chip-roadmap-through-2020). *Windows Central*. Retrieved 2018-10-06.
9. "Always Connected PCs, Extended Battery Life 4G LTE Laptops | Windows" (https://www.microsoft.com/en-us/windows/always-connected-laptop-pcs). *www.microsoft.com*. Retrieved 2018-10-06.
10. "Gigabit Class LTE, 4G LTE and 5G Cellular Modems | Qualcomm" (https://www.qualcomm.com/products/modems). *Qualcomm*. Retrieved 2018-10-13.
11. Furber, Stephen B. (2000). *ARM system-on-chip architecture*. Harlow, England: Addison-Wesley. ISBN 0201675196. OCLC 44267964 (https://www.worldcat.org/oclc/44267964).
12. Haris Javaid, Sri Parameswaran (2014). *Pipelined Multiprocessor System-on-Chip for Multimedia*. Springer. ISBN 9783319011134. OCLC 869378184 (https://www.worldcat.org/oclc/869378184).
13. Kundu, Santanu; Chattopadhyay, Santanu (2014). *Network-on-chip: the Next Generation of System-on-Chip Integration* (1st ed.). Boca Raton, FL: CRC Press. ISBN 9781466565272. OCLC 895661009 (https://www.worldcat.org/oclc/895661009).

14. "Best Practices for FPGA Prototyping of MATLAB and Simulink Algorithms" (http://www.eejournal.com/archives/articles/20110825-mathworks/). *EEJournal*. 2011-08-25. Retrieved 2018-10-08.
15. Bowyer, Bryan (2005-02-05). "The 'why' and 'what' of algorithmic synthesis" (https://www.eetimes.com/document.asp?doc_id=1271261). *EE Times*. Retrieved 2018-10-08.
16. EE Times. "Is verification really 70 percent? (http://www.eetimes.com/author.asp?section_id=36&doc_id=1264922)." June 14, 2004. Retrieved July 28, 2015.
17. "Difference between Verification and Validation" (http://www.softwaretestingclass.com/difference-between-verification-and-validation/). *Software Testing Class*. Retrieved 2018-04-30. "In interviews most of the interviewers are asking questions on “What is Difference between Verification and Validation?” Many people use verification and validation interchangeably but both have different meanings."
18. Rittman, Danny (2006-01-05). "Nanometer prototyping" (http://www.tayden.com/publications/Nanometer%20Prototyping.pdf) (PDF). *Tayden Design*. Retrieved 2018-10-07.
19. "FPGA Prototyping to Structured ASIC Production to Reduce Cost, Risk & TTM" (http://www.design-reuse.com/articles/13550/fpga-prototyping-to-structured-asic-production-to-reduce-cost-risk-ttm.html). *Design And Reuse*. Retrieved 2018-10-07.
20. Brian Bailey, EE Times. "Tektronix hopes to shake up ASIC prototyping (http://www.eetimes.com/document.asp?doc_id=1317504)." October 30, 2012. Retrieved July 28, 2015.
21. Ogrenci-Memik, Seda (2015). *Heat Management in Integrated circuits: On-chip and system-level monitoring and cooling*. London, United Kingdom: The Institution of Engineering and Technology. ISBN 9781849199353. OCLC 934678500 (https://www.worldcat.org/oclc/934678500).
22. Lin, Youn-Long Steve (2007). *Essential Issues in SOC Design: Designing Complex Systems-on-Chip* (https://books.google.com/books?id=7OV9lEn9LiQC&pg=PA176). Springer Science & Business Media. p. 176. ISBN 9781402053528.
23. "FPGA vs ASIC: Differences between them and which one to use? – Numato Lab Help Center" (https://numato.com/blog/differences-between-fpga-and-asics/). *numato.com*. Retrieved 2018-10-17.
24. EE Times. "The Great Debate: SOC vs. SIP (http://www.eetimes.com/document.asp?doc_id=1153043)." March 21, 2005. Retrieved July 28, 2015.
25. "COSMIC" (http://www.ece.ust.hk/~eexu/COSMIC.html). *www.ece.ust.hk*. Retrieved 2018-10-08.

Further reading

- Badawy, Wael; Jullien, Graham A., eds. (2003). *System-on-Chip for Real-Time Applications* (https://books.google.com/books?id=Ha76NqrqPVIC). Kluwer international series in engineering and computer science, SECS 711. Boston: Kluwer Academic Publishers. ISBN 9781402072543. OCLC 50478525 (https://www.worldcat.org/oclc/50478525). 465 pages.
- Furber, Stephen B. (2000). *ARM system-on-chip architecture*. Boston: Addison-Wesley. ISBN 0-201-67519-6.
- Kundu, Santanu; Chattopadhyay, Santanu (2014). *Network-on-chip: the Next Generation of System-on-Chip Integration* (1st ed.). Boca Raton, FL: CRC Press. ISBN 9781466565272. OCLC 895661009 (https://www.worldcat.org/oclc/895661009).

External links

- SOCC (http://www.ieee-socc.org/) Annual IEEE International SoC Conference
- Baya (http://www.edautils.com/Baya.html) free SoC platform assembly and IP integration tool

- Systems on Chip for Embedded Applications (http://www.eng.auburn.edu/~nelson/courses/elec5260_6260/Systems%20on%20Chip%20(SoC).pdf), Auburn University seminar in VLSI
- Instant SoC (http://www.fpga-cores.com/instant-soc/) SoC for FPGAs defined by C++
